Efficient and Reproducible Programming in R

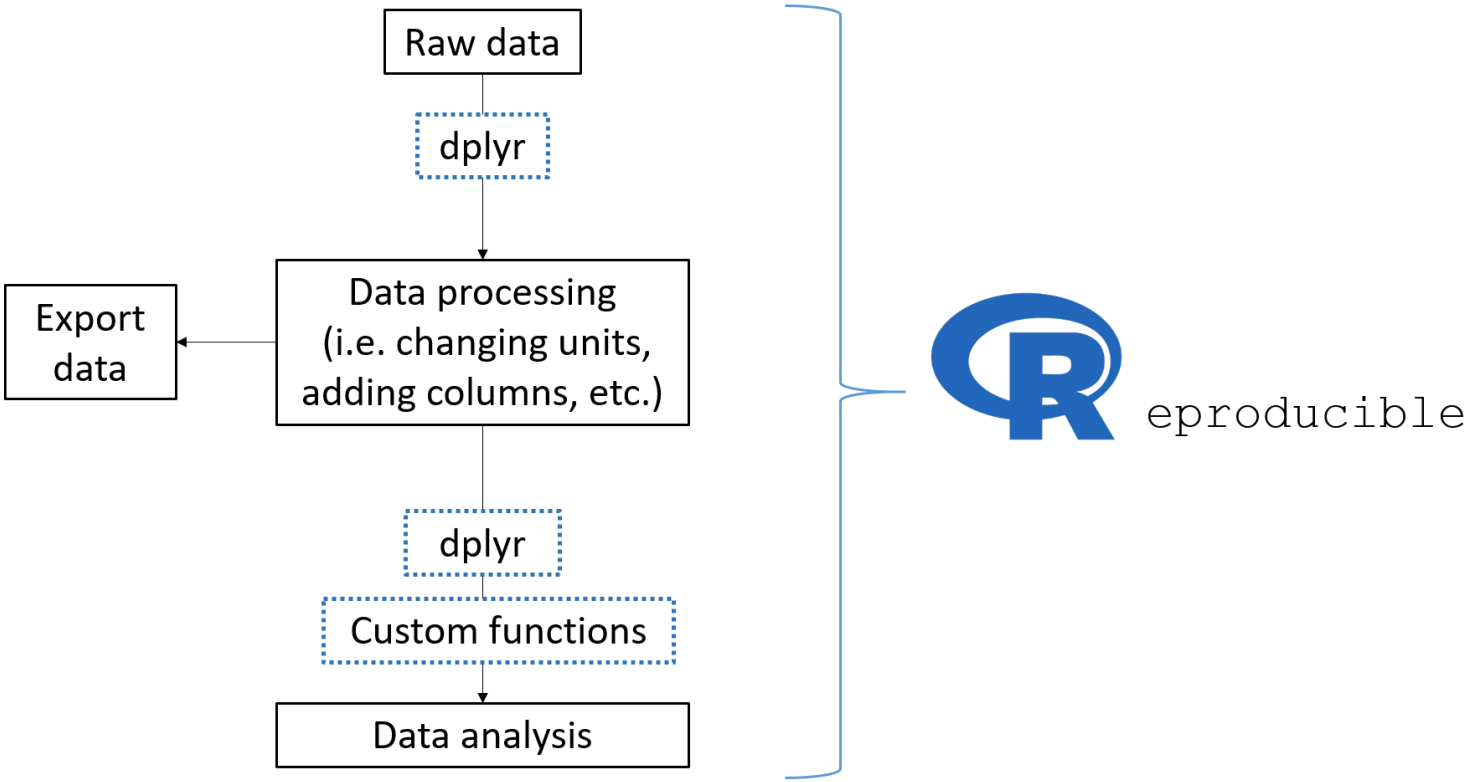
BES Festival of Ecology

Tania L. Maxwell

email: tania.maxwell@inrae.fr
twitter: @tania_maxwell7
website: tania-maxwell.github.io

15-12-2020

Workshop part 1 workflow



@allison_horst

Section 1: raw data

Optimizing your raw data set

How do we set up a data table prior to field sampling or an experiment?

Example: You are setting up an experiment with three plants of corn, and you are measuring the height (cm) at four different time points (T1, T2, T3, T4).

Poll

The data table should contain:

- A) One row per plant (one column for each time point)
- B) One row for each individual measurement (one column for time and one column for height)

Perhaps an "intuitive" data table

Plant	Height_T1_cm	Height_T2_cm	Height_T3_cm	Height_T4_cm
CornA	10	12	15	25
CornB	9.5	11	...	...
CornC				
CornD				

More effective data table for R

Plant	Time	Height_cm
CornA	T1	10
CornA	T2	12
CornA	T3	15
CornA	T4	25
CornB	T1	9.5
CornB	T2	11
...	...	...

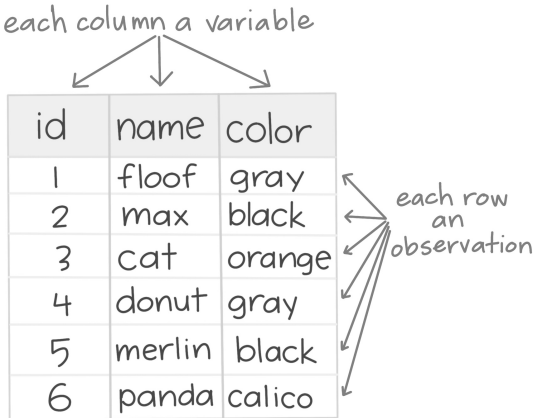
This is what we call 'tidy data'

Introduction to 'tidy data'

> "TIDY DATA is a standard way of mapping the meaning of a dataset to its structure."
>
> –HADLEY WICKHAM

In tidy data:

- each variable forms a column
- each observation forms a row
- each cell is a single measurement



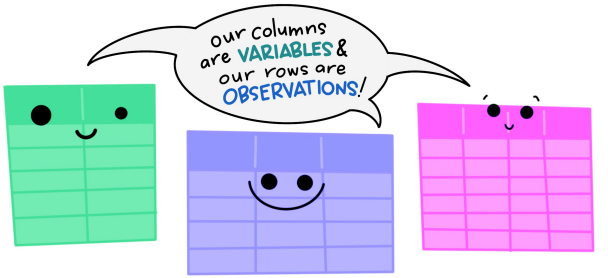
Wickham, H. (2014). Tidy Data. Journal of Statistical Software *59* (10). DOI: 10.18637/jss.v059.i10

@allison_horst

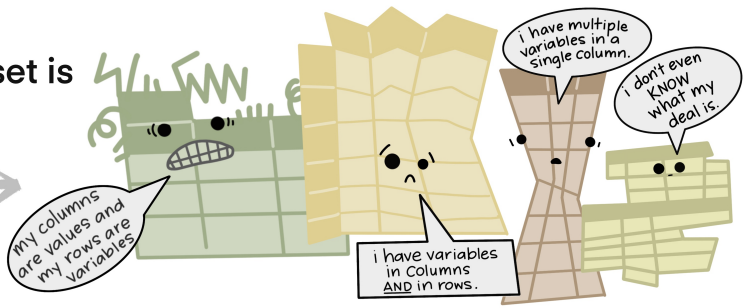
Why use 'tidy data'?

- Most efficient way for R to read your data
- Easier for automation and iteration
- Moving from Excel to R for all calculations : reproducible

The standard structure of tidy data means that "tidy datasets are all alike..."



"...but every messy dataset is messy in its own way."
–HADLEY WICKHAM



@allison_horst

Tidy data: using the `dplyr` package

Section 2: Data processing

Using R instead of Excel

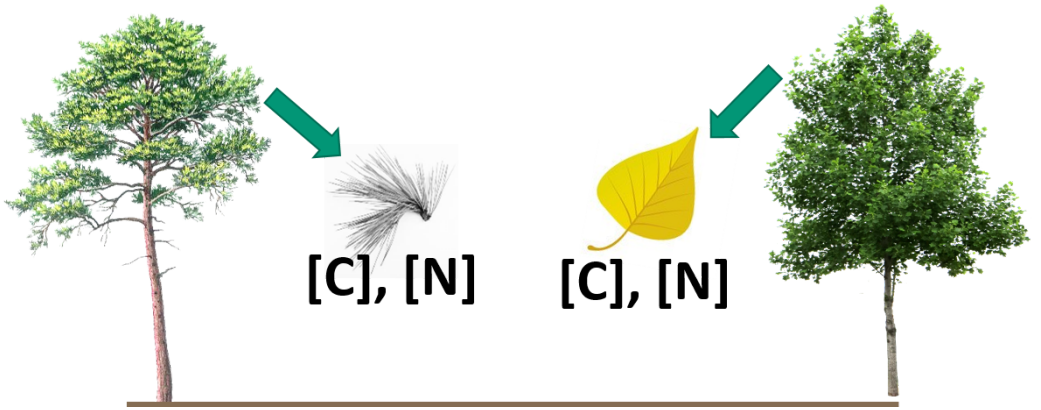
We want to move away from Excel and to R for data processing such as:

- changing units (i.e. multiplying a column by a values)
- create new columns
- reordering the dataset, etc.

Why?

- remove possibility for small errors (i.e. one cell where a function was not applied)
- reproducible code
- easily re-run for different files

Data wrangling with `dplyr`: example



Data structure

```
litter_nutrients18 <- read.table("litter_nutrients_2018.txt", header=T, sep="\t")
litter_nutrients18$Bloc <- as.factor(litter_nutrients18$Bloc) #Bloc as factor
litter_nutrients18$Plot <- factor(litter_nutrients18$Plot, levels = c("B", "P", "B
str(litter_nutrients18)
```

```
## 'data.frame':    24 obs. of  6 variables:
##  $ Species   : Factor w/ 2 levels "birch","pine": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Irrigation: Factor w/ 2 levels "Control","Irrigated": 1 1 2 2 1 1 2 2 2 2 ...
##  $ Bloc      : Factor w/ 6 levels "1","2","3","4",..: 1 1 2 2 3 3 4 4 5 5 ...
##  $ Plot      : Factor w/ 3 levels "B","P","B-P": 1 3 1 3 1 3 1 3 1 3 ...
##  $ C_mg_g    : num  523 517 521 513 524 ...
##  $ N_mg_g    : num  10.4 12.6 12.8 14.2 11.6 ...
```

```
head(litter_nutrients18)
```

```
##   Species Irrigation Bloc Plot C_mg_g N_mg_g
## 1   birch    Control    1    B  523.1  10.42
## 2   birch    Control    1  B-P  517.4  12.64
## 3   birch  Irrigated    2    B  521.1  12.77
## 4   birch  Irrigated    2  B-P  513.0  14.25
## 5   birch    Control    3    B  524.4  11.65
## 6   birch    Control    3  B-P  516.2  10.95
```

Piping: using %>%

Because this data table is 'tidy', we can use `dplyr` for such data manipulations and calculations.

Important Concept : "piping" using %>%

```
New Data Table <- Data Table %>%
                  function1  %>%
                  function 2
                  etc...
```

This reads: "Make a new data table, where you take my data table, apply function 1 to it, and then apply function 2 to the new subset made by function 1 "

Example using `dplyr`

Let's say I am only interested in birch, and I want to calculate a C:N ratio in the fall leaves.

```
library(dplyr)
birch_litter18 <- litter_nutrients18 %>%
  filter(Species == "birch") %>% #only birch
  droplevels() %>% #drop pine level - useful for future graphing
  mutate(C_N = C_mg_g/N_mg_g)
```

- `filter()` is the first function applied to our data frame. Here we choose only birch with the `==` symbol. If we wanted all species except for birch, we could use `!=`

- `mutate()` is applied to the filtered data frame. It creates a new column (here, called `C_N`), which is the division of the `C_mg_g` column by the `N_mg_g` column

Example using `dplyr`

```
birch_litter18
```

```
##    Species Irrigation Bloc Plot C_mg_g N_mg_g      C_N
## 1    birch    Control    1    B  523.1  10.42 50.20154
## 2    birch    Control    1  B-P  517.4  12.64 40.93354
## 3    birch  Irrigated    2    B  521.1  12.77 40.80658
## 4    birch  Irrigated    2  B-P  513.0  14.25 36.00000
## 5    birch    Control    3    B  524.4  11.65 45.01288
## 6    birch    Control    3  B-P  516.2  10.95 47.14155
## 7    birch  Irrigated    4    B  516.1  11.97 43.11612
## 8    birch  Irrigated    4  B-P  518.2  14.40 35.98611
## 9    birch  Irrigated    5    B  522.3  10.16 51.40748
## 10   birch  Irrigated    5  B-P  672.6  16.00 42.03750
## 11   birch    Control    6    B  526.8  14.84 35.49865
## 12   birch    Control    6  B-P  522.3  12.21 42.77641
```

Data wrangling with `dplyr`

Let's say that you want to find the mean per group in your experiment.

Poll

How would you calculate a mean per group?

- A) Excel pivot table
- B) mean() function in R
- C) Sometimes A, sometimes B

Example: means per group

```
litter_means<- litter_nutrients18 %>%
  group_by(Species,Plot, Irrigation) %>%
  summarise(mean_N = mean(N_mg_g)) %>%
  as.data.frame()
```

- `group_by()` is the first function
- `summarise()` is the second function, applied to each Species in each Plot (monocultures vs mixtures) in each water availability treatment
- `mean_N =` names the column in which the `mean(N_mg_g)` is written

```
litter_means
```

```
##   Species Plot Irrigation    mean_N
## 1   birch    B    Control 12.303333
## 2   birch    B  Irrigated 11.633333
## 3   birch  B-P    Control 11.933333
## 4   birch  B-P  Irrigated 14.883333
## 5    pine    P    Control  4.533333
## 6    pine    P  Irrigated  4.643333
## 7    pine  B-P    Control  4.323333
## 8    pine  B-P  Irrigated  4.563333
```

Plenty of examples of `dplyr` functions...

Exporting data from R

```
library(xlsx)
write.xlsx(df, "descriptive_df_name.xlsx", sheetName = "Per_species",
                        col.names = TRUE, row.names = TRUE, append = TRUE)
```

- Name the file with the proper extenstion, i.e. `"descriptive_df_name.xlsx"`
- You can add several sheets to one Excel file, with the `sheetName = "Per_species"` and `append = TRUE` functions
- Similarly, you can use the `write.csv()` function
- Very useful when sharing data with colleagues, or importing the data set into a different script

Section 3: Custom functions

Function syntax

```
fahrenheit_to_celsius <- function(temp_F) {
  temp_C <- (temp_F - 32) * 5 / 9
  return(temp_C)
}
```

- Define `fahrenheit_to_celsius` by assigning it to the output of `function(){}`
- List of argument names within `(parentheses)`
- Body of the function within `{curly braces}` - this is what is executed when you run the function.
- When the function is called , the values we pass to it are assigned to `temp_F` so that we can use them inside the function.
- Inside the function, we use a return statement to send the calculated `temp_C` back.

Function example

```
#calculating 55 degrees fahrenheit to celcius

fahrenheit_to_celsius(temp_F = 55)
```

```
## [1] 12.77778
```

Here, using our function, we see that 55°F is equal to 12.8° C.

Useful ways to use functions

- calculations to be done repeatedly for your data
- replace blanks or . or - with NA values
- quickly create similar figures - i.e. use ggplot and place arguments into the aesthetics

Tips

- Try to give as much information as possible within the function
- Use descriptive value names

References

- R for data science:https://r4ds.had.co.nz/
- https://dplyr.tidyverse.org/
- R cartoons: Allison Horst
- http://swcarpentry.github.io/r-novice-inflammation/02-func-R/index.html
- Other useful websites for using dplyr:
  https://rpubs.com/bradleyboehmke/data_wrangling
  https://seananderson.ca/2014/09/13/dplyr-intro/

I created the slides using xarigan and the R-ladies css.